



realtimepublishers.com<sup>™</sup>

*The Definitive Guide™ To*

# Windows Desktop Administration



*Bob Kelly*

Chapter 8: Asset Management .....	185
Inventory .....	185
Benefits of Maintaining an Updated Hardware Inventory .....	185
Ensuring Minimum System Requirements Are Met.....	186
Knowing Where to Replace Hardware .....	187
Enhancing System Components .....	187
Improving Help Desk Productivity.....	187
Deterring Loss and Theft .....	188
Hardware Inventory Methods .....	188
Software Licensing .....	193
Ensuring License Compliance .....	193
License Types .....	193
Licensing Methods.....	194
Buying Only What You Use.....	195
Software Metering .....	195
Validating Your Environment.....	196
Software Inventory Tools .....	197
Software Inventory Methods.....	198
Change Management .....	201
Establishing Policies for Change .....	202
Goals and Objectives .....	202
Roles and Responsibilities .....	203
Initiation Procedures .....	203
Approval Procedures.....	203
Design, Planning, and Testing .....	203
Scheduling.....	203
Communication.....	204
Recovery and Support.....	204
Documentation and Tracking.....	204
Waivers and Exceptions.....	204
Summary .....	204

## Copyright Statement

© 2003 Realtimedpublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimedpublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimedpublishers.com, Inc or its web site sponsors. In no event shall Realtimedpublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimedpublishers.com and the Realtimedpublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimedpublishers.com, please contact us via e-mail at [info@realtimedpublishers.com](mailto:info@realtimedpublishers.com).

## Chapter 8: Asset Management

Asset management is the tracking of technology assets including hardware and software: knowing where they are, who is using them, and how they are being maintained. Asset management is made up of a combination of policies and procedures that provide you with an invaluable view of your desktop systems. Depending upon your environment, asset management may include any or all of the following topics, which we will cover in this chapter:

- Inventory—Managing and tracking the location and ownership of physical technology assets
- Software licensing—Tracking and managing software licenses
- Change control—Administering the changes to your managed configuration

### Inventory

You may tend to think primarily of software inventory, but you also need to consider physical assets such as computers and their associated components (hard drives, scanners, printers, and so on). In fact, some organizations even include assets such as desks, chairs, phones and other office equipment in the inventory category. Tracking such things may seem irrelevant, and to you, the desktop administrator, they probably are. However, when establishing an inventory system, you should keep in mind that it may be utilized as more than a tool just for the IT department.

Depending upon your environment, it is often valuable to track additional information as it relates to your physical assets including:

- Agreements
- Leases
- Contracts
- Warrantee information
- Maintenance schedules
- Depreciation and other costs, such as repair

### ***Benefits of Maintaining an Updated Hardware Inventory***

Particularly when dealing with software and hardware information, an accurate and up-to-date inventory can help desktop administration efforts in providing valuable data for planning upgrades. Additionally, loss and theft can be more easily identified and reported. When determining which systems would most benefit from replacement and which systems meet certain minimum requirements, accurate computer inventory data is a critical tool. The following sections highlight the benefits of maintaining an update inventory.

## Ensuring Minimum System Requirements Are Met

Every Windows software application provides a minimum and recommended hardware configuration. Outside high-end imaging products and games, it is commonplace to see minimum requirements far below what you would consider a worthy network client. However, unless you have established a workstation lifecycle that rotates old hardware off your network, you may very well find you have client systems in need of software that their hardware cannot adequately support.

In most environments, it is the upgrade to a new version of Windows that dictates the most spiked increase in minimum hardware requirements. Having the ability to quickly identify systems that require updates is a vital step in executing a smooth migration.

Minimum system requirements dictate the software or OS vendors' lowest possible configuration recommendation. Oftentimes these minimum requirements are not enforced, so it is possible for you to perform the installation on lesser systems. Aside from poor performance as a result of such installations, if failures occur, you are not likely to get much help from the product's support department.

Recommended system requirements are usually listed along with the minimum requirements. Whenever possible, it is these recommended requirements that should be your goal to meet (or, hopefully, exceed). As Table 8.1 illustrates, Windows' system requirements have become increasingly demanding with each release.

OS	Processor	RAM	Hard Drive	Video
Windows 95	386DX (486)	4MB (8MB)	55MB	VGA (SVGA)
Windows 98	486/66 (Pentium)	16MB (24MB)	355MB	VGA (SVGA)
Windows NT	Pentium	16MB (32MB)	110MB	VGA (SVGA)
Windows ME	Pentium 150	32MB	320MB	VGA (SVGA)
Windows 2000	Pentium 133	64MB	2GB	VGA (SVGA)
Windows XP	Pentium 233 (300)	64MB (128MB)	1.5GB	SVGA

**Table 8.1: Windows minimum requirements (recommended requirements are listed in parenthesis).**

The Windows Hardware Compatibility List (HCL) contains a list of computer systems and peripherals that have been tested and deemed compatible with Windows. The HCL is often overlooked until problems arise. Particularly if you have added components or peripherals to your systems, take the time to ensure that there are not compatibility issues prior to rolling out a new version of Windows. Compatibility issues have become such a matter of contention since the release of Win2K that Microsoft now provides a database of not only hardware, but also computer brands and software that have passed compatibility testing. You may search the database online at <http://www.microsoft.com/whdc/hcl/default.msp>.

 Because of the Advanced Computer Power Interface (ACPI) capabilities introduced with Win2K, a BIOS update may be required to ensure stability. Failure to install recommended BIOS updates may result in system hardware errors. For more information, see <http://www.microsoft.com/windows2000/server/howtobuy/upgrading/compat/biosissue.asp>.

 A hardware compatibility test kit is provided by Microsoft for hardware manufacturers along with a submission tool and procedures at <http://www.microsoft.com/whdc/hwtest/default.aspx>.

## Knowing Where to Replace Hardware

You have just learned of a new shipment of computers coming in—who will get them? Twenty new machines may mean many more than 20 users receive new computers—many organizations will replace a mid-grade system with a high-grade system, then take that mid-grade system and replace a low-grade system. Often referred to as *cascading*, more users benefit from new systems utilizing this practice. Without sufficient hardware inventory data, it is difficult to identify proper candidates for cascading.

 One situation in which the replacement of systems may become routine is for companies that choose to lease computers for a 2- or 3-year period. This option might be less expensive than the regular purchase of new systems, and (if it is deemed acceptable) the preinstalled OS and software that comes on a new system could reduce administrative efforts in their deployment.

### What to Do with Those Old Computers?

Donating or recycling your old computers and peripherals can not only help the receiving organization, but will also provide your own organization a tax break. When donating, be sure to remove all personal files from the hard drive before you offer a computer for donation. You should also obtain a receipt to document the transfer of ownership. An example where this may become of value is if a computer with a battery is discovered in the trash (which is illegal) you can show that you are not longer responsible for it.

Ensure that you are comfortable with the organization receiving your donation. Ask them to provide proof of nonprofit status before you make your donation. Any legal nonprofit organization or school should be able to provide a letter of request on their letterhead. They should also be able to provide you with copy of their state or federal nonprofit organization registration form.

Share the Technology is an all-volunteer run non-profit organization with a mission is to help salvage recently retired computers while they are still useful, prevent their premature destruction, and get them into the hands of schools, people with disabilities, and nonprofit organizations that can put them to good use. Visit the Share the Technology Web site at <http://sharetechnology.org>.

## Enhancing System Components

Where budgets are limited, it is often beneficial to purchase components such as larger hard drives, additional memory, and even faster processors to replace existing components that may require updating. Unless all the systems in your organization are identical, you will benefit greatly from inventory data to accurately identify which components of which systems should be upgraded. Additionally, inventory data will need to be updated or generated in order to account for the location of these new components in your organization.

## Improving Help Desk Productivity

Gartner Group estimates that as much as 50 percent of the time spent on a Help desk call is used to determine the system's hardware and software configuration. Inventory reports could be used to provide Help desk technicians a detailed report about the troubled PCs configuration quickly.

The more information you have about what's happening on your user's desktops, the more efficiently you can resolve the user's problem. You can speed the troubleshooting process by having the details of the problem system available to you without having to ask the user or examine the system. Additionally, by not having to ask questions, user confidence may also be increased.

If the system reporting trouble has non-standard components and you are aware of these components as a result of an updated inventory, you might be able to look first at the non-standard component as a way to help you to identify a potential cause for the problem. Further, if you identify a problem with a certain device, you can proactively check on other systems that may be at risk. In addition, tracking changes often helps you identify a problem that begins occurring as result of a hardware or software change.

### **Deterring Loss and Theft**

It is impossible to identify and replace lost or stolen hardware without first having some method of identifying the loss. Just the presence of a hardware inventory system may be a considerable deterrent in preventing theft by employees.

#### **Serial Number Tracking**

An automated hardware inventory system may have the ability to contain information that would be difficult or impossible to report on manually. Device serial numbers provide you with the added knowledge of not just which systems have which devices, but exactly which devices. For example, if your inventory reports include RAM serial numbers, you can identify its movement if someone were to take it from one machine and install it in another.

### **Hardware Inventory Methods**

There are many ways to obtain hardware inventory data from a system. In particular, we will discuss the capabilities of WMI and the built-in WinMSD utility to report this information.

In Chapter 7, we discussed the ability of WMI to provide comprehensive system data. Among this retrievable information is a wealth of inventory data; the challenge lies in identifying and pulling meaningful information and displaying it in a user-friendly fashion. Most inventory tools for Windows do just that—collect and provide a clear view of data pulled from WMI. Some key collections you may reference in obtaining WMI data include:

- Win32\_SCSIController—SCSI disk controller information
- Win32\_IDEController—IDE disk controller information
- Win32\_PhysicalMedia—Physical media information
- Win32\_LogicalDisk—Logical disk information
- Win32\_DiskDrive—Disk drive information
- Win32\_Processor—Processor information
- Win32\_NetworkAdapter—NIC information
- Win32\_DesktopMonitor—Monitor information
- Win32\_VideoController—Video adapter information

- Win32\_BaseBoard—Motherboard information
- Win32\_BIOS—BIOS information

 Although many inventory tools today take advantage of WMI to report hardware inventory data, other tools exercise their own proprietary methods to report such information. Later in this chapter, I'll provide a list of several vendors with tools that provide hardware inventory as a feature.

Listing 8.1 shows an example KiXtart script that will display the details of each installed network interface card (NIC). For an example of the output this script would generate, see Figure 8.1.

```
Break On
$Computer = "."
$WMI = GetObject("winmgmts:\\\" + $Computer + "\root\cimv2")
$collection = $WMI.ExecQuery("Select * from Win32_NetworkAdapter",,48)
For Each $item In $collection
? "Adapter Type: " + $item.AdapterType
? "Auto Sense: " + $item.AutoSense
? "Availability: " + $item.Availability
? "Caption: " + $item.Caption
? "Config Manager Error Code: " + $item.ConfigManagerErrorCode
? "Config Manager User Config: " + $item.ConfigManagerUserConfig
? "Creation Class Name: " + $item.CreationClassName
? "Description: " + $item.Description
? "Device ID: " + $item.DeviceID
? "Error Cleared: " + $item.ErrorCleared
? "Error Description: " + $item.ErrorDescription
? "Index: " + $item.Index
? "Install Date: " + $item.InstallDate
? "Installed: " + $item.Installed
? "Last Error Code: " + $item.LastErrorCode
? "MAC Address: " + $item.MACAddress
? "Manufacturer: " + $item.Manufacturer
? "Max Number Controlled: " + $item.MaxNumberControlled
? "Max Speed: " + $item.MaxSpeed
? "Name: " + $item.Name
? "Network Addresses: " + $item.NetworkAddresses
? "Permanent Address: " + $item.PermanentAddress
? "PNP Device ID: " + $item.PNPDeviceID
? "Power Management: " + $item.PowerManagementCapabilities
? "Power Management: " + $item.PowerManagementSupported
? "Product Name: " + $item.ProductName
? "Service Name: " + $item.ServiceName
? "Speed: " + $item.Speed
? "Status: " + $item.Status
? "Status Info: " + $item.StatusInfo
? "System Creation ClassName: " + $item.SystemCreationClassName
? "System Name: " + $item.SystemName
? "Time Of Last Reset: " + $item.TimeOfLastReset
?
Next
```

**Listing 8.1: Example KiXtart script to display the details of each installed NIC.**



```

C:\WINNT\system32\cmd.exe
Adapter Type: Ethernet 802.3
Auto Sense:
Availability: 3
Caption: [00000005] Intel(R) PRO/100 UM Network Connection
Config Manager Error Code: 0
Config Manager User Config: 0
Creation Class Name: Win32_NetworkAdapter
Description: Intel(R) PRO/100 UM Network Connection
Device ID: 5
Error Cleared:
Error Description:
Index: 5
Install Date:
Installed: -1
Last Error Code:
MAC Address: 00:08:02:19:1B:38
Manufacturer: Intel
Max Number Controlled: 0
Max Speed:
Name: Intel(R) PRO/100 UM Network Connection
Network Addresses:
Permanent Address:
PNP Device ID: PCI\VEN_8086&DEV_2449&SUBSYS_00120E11&REV_03\4&11CD5334&0&40F0
Power Management:
Power Management: 0
Product Name: Intel(R) PRO/100 UM Network Connection
Service Name: E100B
Speed:
Status:
Status Info:
System Creation ClassName: Win32_ComputerSystem
System Name: PACKAGEBOX1
Time Of Last Reset: 20030715043414.000000-240

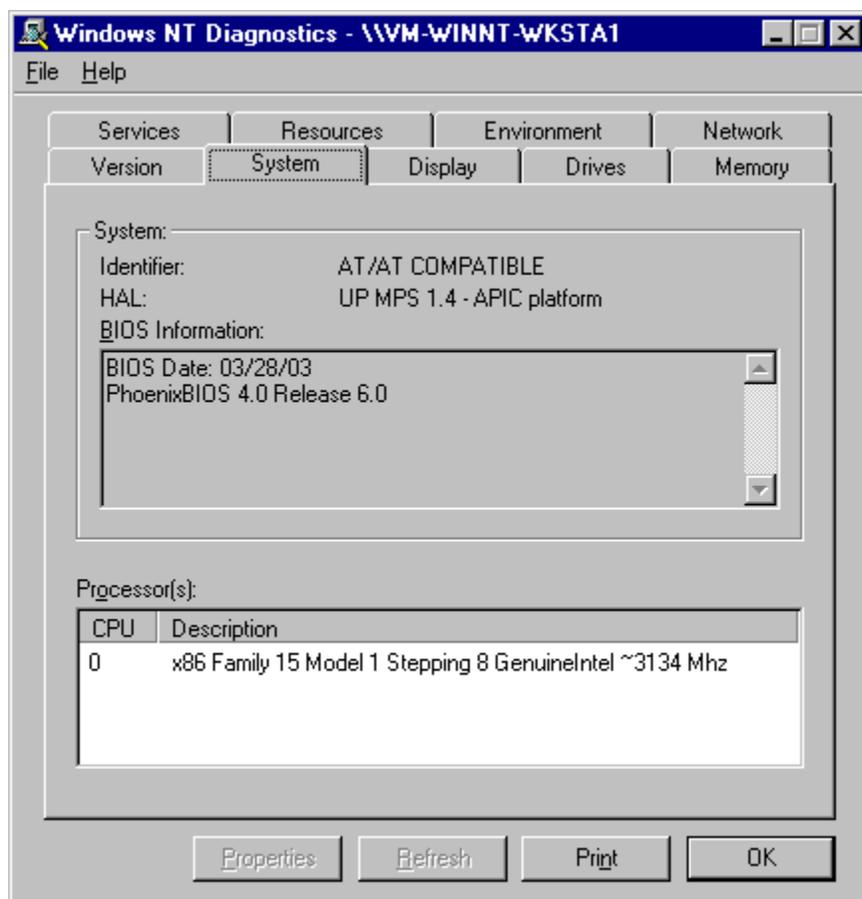
```

**Figure 8.1:** Example of network information script output using WMI.

Another way to gather system details in an automated manner is to utilize the WinMSD tool. WinMSD has been updated through the past few versions of Windows. For NT, WinMSD can specify the UNC path to a system and utilize any of the following command-line switches:

- /a—All details
- /s—Summary details only
- /f—Send output to a file <computername.txt> in the current directory
- /p—Send output to a printer

Run alone, NT's WinMSD will launch in GUI mode, as Figure 8.2 shows.



**Figure 8.2:** WinMSD for NT in GUI mode.

The Win2K incarnation of WinMSD (msinfo32) also provides a great deal of detail about the specified system. You can use the following command-line parameters to specify specific sections to output:

- /msinfo\_file is used to open a specific NFO or CAB file
- /nfo or /s will send output to a specific NFO files
- /report directs text-formatted output to a specific file
- /computer connects and displays information regarding a specific remote system
- /categories displays the specified categories (used to limit output)
- /category sets the focus of the GUI display to the specified category at startup



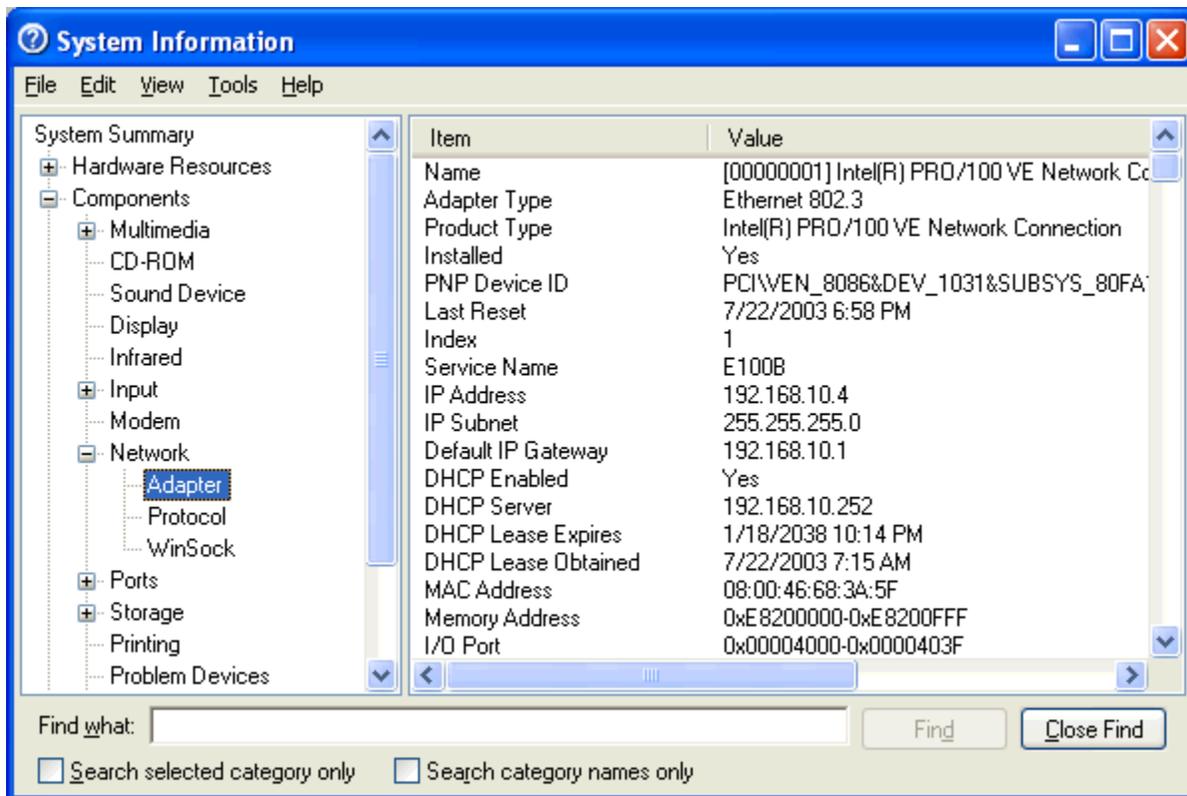
An NFO file is System Information File, which is a binary file that may be displayed using the System Information tool (msinfo32.exe) as a viewer.

You can use the following command to return all available information, which generates a detailed report in a text file named `inventory.txt`:

```
WinMSD.exe /report c:\inventory.txt
```

 For more information about Win2K's WinMSD tool, see the Microsoft article "Windows 2000 Command-Line Parameters for Msinfo32.exe."

Figure 8.3 shows the Windows XP WinMSD tool interface.



**Figure 8.3: Windows XP WinMSD (msinfo32.exe).**

 You can use the new `msinfo32.exe` filename or the older `winMSD.exe` filename to initiate this tool on Win2K and Windows XP systems. However, on Windows XP, you must call `MSInfo32` directly to utilize the supported command-line switches.

 When automating WinMSD from the command line, be aware that if the Server Service is not running, WinMSD will show a warning dialog box.

## Software Licensing

When many think of inventory and asset management, software inventory immediately comes to mind. Do you have enough licenses? Do you have too many? What software do you have installed throughout your network? We will cover these and other topics as they relate to software licensing in this section.

### **Ensuring License Compliance**

Current legislature supports software abuse policies and heavy fines are now being imposed upon companies not compliant with software licenses. Failure to maintain a sufficient number of licenses on your network can be a costly situation.

More and more employees are utilizing applications on desktops and laptops. Take into consideration the number of licenses purchased, various software versions operating within your company, and the types of licenses you own for each application. With these and other variables, software license compliance tracking can be a challenge—even with the right tools. Operating without such tools can make license management a nearly impossible, and certainly time-consuming, task.

### **License Types**

In addition to the many ways licenses may be enforced, there is also the matter of how licenses are to be allocated—some are to be tracked per user, others are to be tracked per seat (system), there are licenses intended for a limited number of active users, and finally some cover an entire organization.

- Per user—Licensing tracked per user or per developer means that you must have a license for each individual who makes use of the software. Per user licensing normally allows for multiple installations so long as the number of individuals that use the software is accounted for.

 Software licensed per user normally records licensing information in the user profile. In this way, the user will be able to use the software in more than one location, but other users attempting to use the software will be prompted for registration information and/or denied access if their user profiles do not contain this valid licensing data. Although the licensing information is sometimes stored in a file, it is more commonly found in the registry under the vendor's software key (for example, in the HKEY\_CURRENT\_USER\Software\\<AppTitle> key).

- Per seat—Per seat licensing is typically reserved for client/server applications for which many clients throughout a network have a client-side service or agent installed. Per seat commonly means per system; thus, regardless of the number of users on the network, each computer that executes the application must have its own license.

Per seat licensing is usually enforced by the server-side component of the software, such as with many desktop management systems. The server counts the number of client systems that it has worked with and tracks this number against the number of purchased licenses. It is up to the vendor as to how license violations are to be handled; some will provide warnings, while others will cease operation (at least on any additional systems) until the number of per-seat licenses is appropriately increased.



Particularly for server software products, an increasingly common licensing model is per processor, with which licenses are required for each CPU in the server.

- **Concurrent usage**—Concurrent licensing refers to the number of active instances at any one time. In this scenario, you may have only a dozen licenses for a particular application though you have hundreds of installed systems and potential users. Not until the thirteenth user of the software attempts to initiate the application do you have a license violation.

Concurrent usage is often tracked by organizations using software metering tools. Metering tools record the times a software product is launched and exited. Once the maximum available licenses are in use, some metering products will even go so far as to halt a license violation from taking place by denying further execution of the application until other instances on the network are closed.

- **Site licenses**—A site, or enterprise, license indicates that an entire company, domain, or location can utilize the software without concern for the number of systems, users, or its concurrent usage. Although the most expensive, such a license is also the easiest to administer.

### **Licensing Methods**

There are several ways that software vendors dictate licensing requirements for their software. As we will cover in this section, these methods may include the implementation of license management servers, registration codes, and files.

- **License servers**—A license server is usually used to track concurrent usage of a product. Per seat licensing is often tracked by the application itself, whereas a license server will actually do the job of recording the number of instances currently in use. When the number of purchased licenses is exceeded, further attempts to launch the application are typically met with a warning dialog box that instructs the user of the problem, then exits the application. Such being the case, you must either wait for licenses to become available (as others close the application) or purchase additional licenses to increase the number of allowable instances.
- **License files**—A typical license file implementation is in the format of a binary file provided when a purchase is made. The license file may include information such as the number of per seat licenses, concurrent licenses, company name, expiration date, and product support or upgrade coverage. The application provides a method for accepting this license file where the details of your purchase may be processed by the application in question.
- **Serial/Registration strings**—The most common implementation of software registration is for applications to request a serial number, registration, or product code during installation or upon first launch of the software. Many applications are usable in a limited configuration until such registration information is provided to “unlock” the application and allow full use of what the program has to offer. This setup provides vendors the ability to let you try the software in a limited capacity and/or for a certain period of time. Further, it saves you from the need to redeploy the software if the evaluation results in the purchase of licenses.



 Although some licensing implementations use information—such as the network card MAC address or other unique identifier—to pair a provided license code with a specific system, the same license code may often be used on multiple systems. Is this okay? To know for certain, you would need to ask each software vendor that you deal with. However, it is often considered to be acceptable to deploy software using the same registration information as long as you have the proper number of licenses in your possession.

### **Buying Only What You Use**

Having a clear picture of what you have and what you are using can save a considerable amount of money when it comes time to purchase upgrades or extend support offerings. If software is deployed to all systems, and you are able to determine that only 10 percent of the users ever utilize it, renew the annual support contract for only those systems that require it. How do you know what is being used? Through software metering.

### **Software Metering**

Software metering measures concurrent usage of applications on your network. This measurement may be limited to certain managed applications or may include all software across the enterprise. At its most basic implementation, metering software measures when the software is launched and when it is closed. This information alone can be very valuable when presented in concise reports. However, in addition to reporting, software metering solutions may also manage and enforce concurrent usage licensing limits. Simply collecting and reporting information about software usage is known as *passive metering*, license enforcement is referred to as *active metering*.

- Passive metering—More common to metering solutions is the offering of passive metering functionality. With this functionality, executions are monitored and reported upon. Some solutions may generate administrative alerts when licenses are exceeded; however, these alerts are usually not provided in real time. Usage data is often collected locally, then reported to a management server at a defined interval. When collated, or when reports are generated, any violations can be identified and licenses may then be purchased to remedy the situation.

 Depending upon the metering solution, usage data can simply be execution and exit times or it may actually track activity. The ability to see the difference can be helpful because people may simply leave a program minimized all day and not actually use it. Although knowing this behavior does not necessarily help with licensing, it does give you a more accurate picture of what is really being used in your environment. Among other things, you can use usage data to determine what training may be of value as well as assessing which software upgrades may be of the most value to your users.

- Active metering—License management servers are configured to track the execution of applications on the network. Although this tracking might be limited to certain applications, when launched, the client software checks with the licensing server to determine whether there are available licenses. If there are, the application is allowed to execute and the number of available licenses is decremented. When the number of configured licenses is exceeded, the metering solution may notify the user to try again later or may even provide the option to be notified when a license becomes available.

Naturally, the process of checking a license server for available licenses prior to allowing or disallowing execution can be a performance hit. In fact, there are a shrinking number of products providing this capability as a result of this performance hit. However, some implementations may be faster than others, and depending upon your network capabilities and licensing needs, this may be a suitable solution.

 Although software metering might seem like an excellent solution to licensing problems, you would be wise to investigate your software license agreements to determine whether this is acceptable. If per seat licensing has been dictated by the vendor, it may be unacceptable to implement your own concurrent usage management.

### ***Validating Your Environment***

Suppose you have deployed a new software package to 300 systems. How many copies are out there? If there is less, you can use the information to help investigate possible failures. If there are more, you can use the information to help investigate unauthorized installations (or installations performed outside your deployment process).

Although many deployment solutions provide a means of reporting successes and failures, many do a questionable job of it. Is it installed because the command line used to perform the silent installation did not return an error? Is it installed because the shortcut is now there or because some other file you previously identified is now present? However you go about it, validating your deployment efforts using software inventory provides added assurance of your success.

Perhaps the software you are to deploy is an update to an existing application. If some do not have the previous version as you expected, your installation will need to take that condition into account. Further, there may be a very early version of the software on the network that you may not have otherwise accounted for. Examining software inventory reports for dependencies or prerequisites can help you to provide a deployment package catered to the conditions that exist within your unique environment—software inventory can show you what that unique environment really is.

Having established a process for software deployment, there will still be those that step outside that process. Either to side-step the approval process or to simply get what they want faster, manual installations can be a real concern. Comparing what people have been deployed to what they actually have can provide a picture of where these process violations may exist.

Software inventory can also tell you who is not playing by the rules. By comparing (manually or programmatically) reports that show what has been assigned to a machine via your change management process and reports that show what is actually installed, you may identify unmanaged software (for example, you can discover how many games are installed on your network).

## Software Inventory Tools

There are several software inventory tools available that offer many features. Some features to watch out for include:

- Web reports—for reporting, web reports have quickly become the preferred method for many organizations. With web reports, no client software is required to view them and they may be easily shared or utilized on an existing corporate intranet.
- Report access control—you may not want everyone to access these reports. Some inventory and usage tracking tools provide a means for users to see a report of their own system, but not that of others. When access is limited, learn how it is limited (group membership, account privileges, separate username and password, etc.)
- Software requirements—some solutions do not require a client application to be deployed. If it does require a client service, what deployment mechanisms are offered by the vendor?
- Assurance of licenses for key personnel—if the president of your company cannot launch his PowerPoint presentation, you do not want fingers pointed at you!

Table 8.2 provides more information about the features provided by some of the popular software inventory and metering products (this table is not an all-inclusive list of products).

Company	Product	Software	Hardware	Active Metering	Passive Metering
ABC Systems	LAN Licensor			X	X
Alloy Software	Network Inventory Navigator	X	X		X
Altiris	Client Management Suite	X	X		X
Executive Software	SiteKeeper	X	X		
Express Matrix	Express Software Manager	X	X	X	X
Globetrotter Software	SAM wrap	X		X	X*
iInventory	LAN auditor	X	X		
Integrity Software	SoftTrack	X		X	X
Microsoft	Systems Management Server	X	X	X	X
NetSupport	NetSupport TCO	X	X	X	X
Sassafras Software	KeyAuditor	X	X	X	X

Company	Product	Software	Hardware	Active Metering	Passive Metering
Scalable Software	Survey Suite	X	X		X
Software Innovations & CPS	SystemHound	X	X		
Tally Systems	TS.Census	X	X		X*
Tangram Enterprise Solutions	Asset Insight	X	X	X	X
Unipress Software	FootPrints Asset Management	X	X		

**Table 8.2: Software inventory and metering products (\* = requires optional add-in or module).**

### Software Inventory Methods

There are several methods that you can employ to determine which software is installed on a system. Through COM, the WindowsInstaller object can report on not only what is installed, but also the state of the installation itself. Listing 8.2 shows a KiXtart script that displays the installation state for each Windows Installer–managed application on the local system.

 For more information about the ProductState property of the WindowsInstaller object, check out [http://msdn.microsoft.com/library/en-us/msi/setup/installer\\_productstate\\_property.asp](http://msdn.microsoft.com/library/en-us/msi/setup/installer_productstate_property.asp).

```

$MSI = CreateObject("WindowsInstaller.Installer")
$Products = $MSI.Products
For Each $Product In $Products
  $StateVal = $MSI.ProductState($Product, "ProductName")
  Select
    Case $StateVal = 0  $StateTxt = "Broken"
    Case $StateVal = 1  $StateTxt = "Advertised or Removed"
    Case $StateVal = 2  $StateTxt = "Absent"
    Case $StateVal = 3  $StateTxt = "Local"
    Case $StateVal = 4  $StateTxt = "Source"
    Case $StateVal = 5  $StateTxt = "Default"
    Case $StateVal = -1 $StateTxt = "Unknown"
    Case $StateVal = -2 $StateTxt = "Invalid Argument"
    Case $StateVal = -4 $StateTxt = "Source Absent"
    Case $StateVal = -5 $StateTxt = "Incomplete"
    Case $StateVal = -6 $StateTxt = "Bad Configuration"
    Case $StateVal = -7 $StateTxt = "Not Used"
  EndSelect
  ? $MSI.ProductInfo($Product, "ProductName") + " is " + $StateTxt
Next

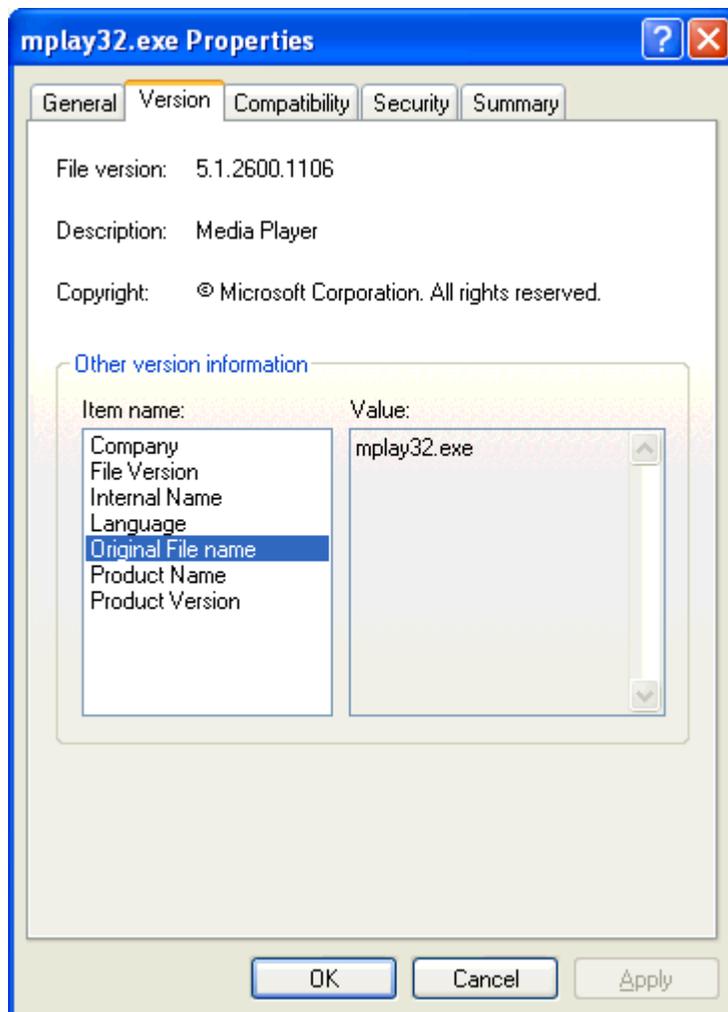
```

**Listing 8.2: An example KiXtart script that displays the installation state for each Windows Installer–managed application on the local system.**

Because the Windows Installer method will only report those applications managed by the Windows Installer service, scanning the target system for files is a more thorough means of determining which software is present. This scan can be a time-consuming process, but can be very accurate. Some inventory products scan the file system and compare it with a proprietary database that identifies the installed software. Other solutions enumerate the files of the OS and examine their properties for information (see Figure 8.4) and some may use a combination of these two methods.

Using KiXtart's `GetFileVersion` function, you can obtain the informational data for any specified file. When present, the following file properties can be returned. Keep in mind that all, some, or none of the file version information may be present for any given file. The following descriptions represent the intended use for each possible field; however, the developer is free to include any information desired. It is this same process that some software inventory products utilize to identify files:

- **Comments**—Returns any specified comments for the file. This property is usually used to provide additional information that can be displayed for diagnostic purposes.
- **CompanyName**—Returns the name of the company that produced the file.
- **FileDescription**—Returns a string used to describe the file. This string may be used to dynamically produce a description along with (or in place of) the associated file name, which can sometimes be difficult to interpret. Unfortunately, this field is not commonly used.
- **FileVersion**—Returns the version of the file.
- **InternalName**—Returns the file's internal name, if one exists. For example, this string might contain the module name for a Dynamic Link Library (DLL), a virtual device name for a Windows virtual device, or a device name for an MS-DOS device driver.
- **Language**—Returns the full English name for the file's language.
- **LegalCopyright**—Returns all listed copyright notices, trademarks, and registered trademarks that apply to the file. This property may include the full text of all notices, legal symbols, copyright dates, trademark numbers, and other copyright information.
- **LegalTrademarks**—Returns all trademarks and registered trademarks that apply to the file. This property may include the full text of all notices, legal symbols, trademark numbers, and other trademark information.
- **OriginalFilename**—Returns the original name of the file, not including a path. This property can be used to determine whether a file has been renamed from its original file name. If the file is specific to a non-FAT file system, this name might not be in the standard MS-DOS 8.3 naming format.
- **PrivateBuild**—Returns by whom, from where, and why this private version of the file was built.
- **ProductName**—Returns the name of the product with which this file is distributed.
- **ProductVersion**—Returns the version of the product with which this file is distributed.
- **SpecialBuild**—Returns how this version of the file differs from the normal version.



**Figure 8.4:** File properties displayed for Microsoft Media Player.

By examining the OriginalFileName property, inventory solutions can identify files that may have been renamed by a user in order to hide them.

Another software inventory method is to use the Add/Remove Programs Control Panel applet, which lists the software installed on a machine that support an uninstall process. This list accounts for a great percentage of the software you are likely to have in your environment. By reading the values used by the Add/Remove Programs applet in the Uninstall key of the registry, you can obtain the names of each application installed on a system (and in many cases, much more).

See Chapter 7 for more information and sample scripts.

## Change Management

In Chapter 2, we discussed establishing and deploying your initial baseline configuration. In Chapter 4, we discussed deploying software and updates to those systems. Remember that all the benefits of a baseline are quickly lost if you don't manage the changes that must occur throughout a system's *life cycle*. A system's life cycle covers its planned deployment, maintenance, and eventual replacement.

Suppose you have just rolled out the best documented, well-tested, and rock-solid baseline ever conceived. Thinking that it can stay that way very long is a big mistake. You must plan for the inevitable changes that will follow. First, let's discuss some of the reasons changes may be instigated in your environment:

- User requests—Aside from requests for new or upgraded software and the “My computer is slow, I need more RAM” request, consider that you may also need to account for evaluation software requests. Particularly with evaluations, it is important to have a plan for the inevitable removal, or even the possible in-place activation, of the software at the end of the evaluation period. Testing the removal of software becomes a step of increased importance in this situation. Hopefully, the process put in place will keep you from having to be the one to deny these regular user requests.
- OS upgrades—Some day, someone will need to upgrade to a new OS. Sometimes it is difficult to look so far ahead, but in the world of Windows, Microsoft has made it easier to see that day by publishing its Windows Product Lifecycle.

This life cycle states that Windows desktop OS licenses will be available for purchase for a minimum of 4 years, with assisted support offerings available for up to 7 years after general availability. During these periods, Windows desktop product assisted support will move through a modified set of Product Lifecycle Phases. Standard support offerings will be available for the first 5 years after general availability. It will then go into an Extended Support Phase between 5 and 7 years after general availability. During this time, assisted support on an hourly basis and hotfixes may be available. However, there will be no requests for warranty support, design changes, or new features handled by Microsoft during this extended phase. Table 8.3 shows the Windows Product Lifecycles phases by date.

Desktop OS	Date of General Availability Start Date	OEM and Retail License Availability End Date
MS DOS 6.xx	1-Jun-94	31-Nov-01
Windows 95	15-Aug-95	31-Dec-00
Windows NT Workstation 4.xx	29-Jul-96	30-Jun-02
Windows 98	30-Jun-98	30-Jun-02
Windows 98 SE	30-Jun-99	30-Jun-02
Windows Millennium Edition	31-Dec-00	31-Dec-03
Windows 2000	31-Mar-00	31-Mar-04
Windows XP	31-Dec-01	31-Dec-05

**Table 8.3: Windows Product Lifecycle dates.**

 For more information about Windows Product Lifecycles, visit <http://support.microsoft.com/lifecycle>.

- Software upgrades—Software release schedules can be annual, quarterly, or even monthly. Patches and fixes can be even more frequent. Although not all software will be critical enough to warrant this regular attention, much will. You must also consider the fact that if your software becomes too outdated, support and compatibility issues may also drive change. Plan for the likelihood of upgrades and patches and you may see opportunities to prepare for it when the time comes.
- Fixes—In particular, security fixes have been a common hot button for many environments. With Windows being scrutinized and taken apart by so many, security vulnerabilities surface on a frighteningly frequent basis.
- Hardware upgrades—As discussed earlier, increasing recommended requirements can be expected at unspecified intervals. For years, few software packages required more than a 386-based processor and varying amounts of RAM. Today, increases in hardware requirements, particularly in drive space and RAM, are increasingly frequent. Conversely, hardware upgrades may result in a need for software updates as well (as in the case of drivers and hardware support software).

 Some organizations have an established hardware lifecycle in place. A percentage of the user desktops in the organization are retired and replaced at a regular interval. For example, 33 percent of the machines are replaced every year in a 3-year life cycle.

- Business changes—Reorganizations and relocations represent another common catalyst for change. For example, a newly formed group of people may need a specialized baseline or may have increased hardware requirements.

### ***Establishing Policies for Change***

The basis for a change management process is established policies and procedures. It is important to have a wide organizational involvement in the establishment and enforcement of these policies. These policies and procedures should be refined as improvements can be identified.

### **Goals and Objectives**

What are you trying to accomplish with your change management process? In an attempt to attain a consistent environment, one mindset is to establish and maintain a common baseline by updating all systems together. Installing all software on all systems may seem like a licensing problem, but a software metering tool can provide a solution. Many organizations simply want to ensure that configurations are documented and managed in a controlled manner so as to minimize license violations and more easily associate specific problems to similarly configured systems. For example, if you know two software packages have an issue running on the same system, you can be proactive in addressing the problem if you can quickly establish which systems are at risk.

## Roles and Responsibilities

There may be a great number of people involved in your change management process; thus, it is important to establish clearly defined barriers of responsibility. Roles may include making decisions or even just sitting in on meetings to represent a division or group.

## Initiation Procedures

Depending upon the environment, requests could be initiated by a Web form, a board presentation, or a simple email. It is important that users should know where to go to request action and that a summary of the approval, development, testing, and deployment procedures be available.

## Approval Procedures

Although in some organizations, it may seem like everything gets approved, there should be an approval process as a vital step in your change management process. Define what is required for approval or what factors will be weighed.

### Establishing Business Reasons for Change

Installing the fifth different spreadsheet program on the network? Before software is ever purchased, there should be a business need presented for why the new software is required. The same goes for upgrades—the fact that an upgrade exists is not always the best reason for deploying it.

If the new software requested provides new functionality needed by your users, that reason is a good basis for a business case. An upgrade that makes a software product faster is a good basis for a business case. If the new or upgraded software can improve the work your users need to perform, this improvement can often be converted into time, and thus, into money.

## Design, Planning, and Testing

As discussed in Chapter 4, establishing a process for designing, planning, and testing is essential. Not just in the implementation of new software and updates, but in hardware changes as well. How can the job be done? What is the best way to do it? And what could go wrong? After you answer these questions, validate the decision with testing in an environment as close to production as possible.

## Scheduling

As mentioned earlier, having representatives from the various departments or business groups within your organization involved in meetings about change decisions is a very good idea. A representative from another department will have more intimate details of how changes may adversely affect their operations—in particular, schedules. Can the deployment happen at night? Over the weekend? Must it happen during business hours? If you need a user present, the decision is clear. If the change can be automated to occur during off hours, consider the possible affect of not having anyone present to address unexpected problems.

## Communication

Are you going to announce deployments ahead of time? Some feel that advertising deployments causes users to look for problems where they may not exist. A typical example is for users to blame their computers' slowness on something the deployment team must have done to their machines. Despite this potential drawback, it is crucial to keep users informed. Letting users know of changes before they occur allows them to be aware of your successes and not just the failures. If users only learn of changes that affected them adversely, they will be justified in looking down upon the capabilities of the deployment team. What can the users expect? Will their machine reboot? Will the software they use behave differently? These are important topics to consider in change management.

## Recovery and Support

Planning for problems and how to deal with what may go wrong beforehand can pay off if issues arise. Any change to be implemented should require a backout plan.

As a whole, a support process is typically a rather involved one. As it pertains to change management, think specifically how support will be implemented for the change. In some cases, it may make good sense to have the Help desk remain the primary focal point for support, but with instructions to route any problems related to the changeover to the team implementing the change.

 We explore support in more detail in Chapter 6.

## Documentation and Tracking

Without documenting and tracking changes, maintaining control of your environment can quickly get away from you. Although software inventory can give you good insight into how things are, documentation will give you a picture of how things should be.

## Waivers and Exceptions

No matter how streamlined your process, there will always be situations in which things must happen with greater speed. Establish a process for waivers and exceptions as well as an authoritative group for approving these exceptions. Although it is important to be ready to handle exceptions, it is even more important to avoid abuse. When every change request is classified as an urgent emergency, the benefits of the work done to establish a process for change may quickly diminish.

## Summary

In this chapter, we have discussed software and hardware inventory as well as the benefits and methods that you can employ to maintain them. We covered the benefits of software license management as well as a number of tools available and features to consider when evaluating solutions. Finally, we covered change management, which should be developed and maintained from before the first workstation is even rolled out.